An Architecture for Semi-Automatic Collaborative Malware Analysis for CIs

Giuseppe Laurenza, Daniele Ucci, Leonardo Aniello, Roberto Baldoni
Research Center of Cyber Intelligence and Information Security
Department of Computer and System Sciences "Antonio Ruberti", "La Sapienza" University of Rome
{*laurenza,ucci,aniello,baldoni*}*@dis.uniroma1.it*

***Abstract*—Critical Infrastructures (CIs) are among the main targets of activists, cyber terrorists and state sponsored attacks. To protect itself, a CI needs to build and keep updated a domestic knowledge base of cyber threats. It cannot indeed completely rely on external service providers because information on incidents can be so sensitive to impact national security. In this paper, we propose an architecture for a malware analysis framework to support CIs in such a challenging task. Given the huge number of new malware produced daily, the architecture is designed to automate the analysis to a large extent, leaving to human analysts only a small and manageable part of the whole effort. Such a non-automatic part of the analysis requires a range of expertise, usually contributed by more analysts. The architecture enables analysts to work collaboratively to improve the understanding of samples that demand deeper investigations (intra-CI collaboration). Furthermore, the architecture allows to share partial and configurable views of the knowledge base with other interested CIs, to collectively obtain a more complete vision of the cyber threat landscape (inter-CI collaboration).**

***Keywords*-collaborative analysis; automatic malware analysis; critical infrastructure protection;**

I. Introduction

Malware are evolving, new families are being continuously developed, and the campaigns they use to spread are getting more frequent and effective, leading to impressive numbers of unknown samples being discovered every day [1].

The most famous example of cyber attack to a CI is *Stuxnet* [2], a malware designed specifically to damage Iran's nuclear facilities. This malware targeted the PLCs that automated the electromechanical processes, with the aim of gathering private information on the facility and damaging fast-spinning centrifuges. One of the most recent cyber attack against a CI was the so called *BlackEnergy* [3]: "On December 23rd, 2015, around half of the homes in the Ivano-Frankivsk region in Ukraine (population around 1.4 million) were left without electricity for a few hours". This attack was launched by using a malware that damaged some critical components of Ukraine energy companies. The number of attacks CIs face are on the average one every a few seconds. Even though many of these attacks are blocked and filtered out by automatic detection mechanisms, some of them have to be brought to the attention of cyber analysts because they could be new form of malware that have to be carefully manually inspected.

Let us remark that a CI is a complex system deployed over a (sometimes) large geographical territory that ensures a 24/7 service towards citizen. Thus it embeds a huge number of vulnerabilities that can be exploited by attackers. Many of them are known to the CI, however it is sometime impossible to patch them due to service interruption or because the removal of a vulnerability could take years (e.g. replacement of millions of hackable smart meters already deployed). Hence, vulnerabilities and successful attacks are probably the most sensitive information of a CI that could severely impact its reputation and national security. As a result, fully relying on external "security providers" such as antivirus and intrusion detection systems companies is not a good option for CIs because these companies would manage most of such confidential information with risk of leakage.

Ideally, in case of a hack event, the targeted CI would need to keep the story as much confidential as possible, share the information with national authorities in charge, carry out all the investigations required to shed light to what happened, and finally decide what information to disclose and how to communicate it to the public. The more external actors are involved in such critical phase, the highest is the risk of information leakage. As a consequence, CIs need to establish internal processes to manage their cyber issues, and a key element to achieve that consists in the capability of examining the behavior of malware samples.

CIs would need to keep an always updated domestic knowledge and awareness on novel malware samples. This can be obtained through a proactive approach and by establishing collaborations with third parties. Given the huge amount of samples to analyze, the analysis process should be automated as much as possible. Where automation is not possible, such as when an unknown form of malware is detected, human intervention by some internal analysts is required. To be even more effective, such manual analysis should be done collaboratively by experts having skills in diverse areas, so as to widen the coverage of the aspects to investigate and consequently enrich the domestic knowledge. Finally to increase the domestic knowledge, a controlled form of information sharing is necessary with national/international agencies and other CIs and organizations facing similar cyber threats [4], [5].

In this paper we propose an architecture for a framework for CIs dedicated to malware analysis, which takes into

account (i) the need of CIs for high confidentiality of their cyber security concerns, (ii) the capability of analyzing very large volumes of malware samples in a mostly automated way, while still enabling analysts to contribute, (iii) the possibility to account for *intra-CI collaboration* by allowing different human operators to add results to a same sample analysis, and (iv) the opportunity to open to *inter-CI collaboration* by sharing selected subsets of developed knowledge with other CIs. Although the architecture we designed is highly tailored to meet the confidentiality requirements typical of a CI, its field of application can be broader. As an example, also enterprises and corporations could surely benefit of such a solution.

Section II reports on the state of the art about malware analysis tools, with focus on automated gathering, elaboration and reporting of samples. Section III defines precisely the requirements of such a framework, and Section IV describes an architecture able to meet them. Conclusions are drawn and future works are presented in Section V.

II. Related Work

Several works in literature addressed a number of facets related to automatic malware analysis and detection. Most of them focus on specific aspects and don't engage in designing a general architecture for automatic malware analysis.

As an example, the following two papers deal with automatic static analysis and detection. In [6] a framework is presented for automatic static malware analysis based on control flow graphs and behavioral signature matching, which obtains good results for real-time detection latency. [7] proposes an automatic framework for detection based on data mining, which uses instruction sequences extracted from samples to generate malicious patterns to be used at runtime to detect if a sample is a malware.

MAIL [8] is an intermediate language to express assembly code so as to ease the detection of metamorphic malware. Authors claim this is a valuable step towards the automation of malware detection, so this work actually addresses a specific piece of the overall problem we are tackling.

Other works explicitly focus on the automatic dynamic analysis rather than on the detection, and thus again only deepen a particular aspect. An architecture for automatic malware analysis is presented in [9], where an emulator is used to analyze samples taking advantage of dynamic and taint analysis techniques. [10] carries out dynamic analyses in sandboxes and employs machine learning techniques.

AMAL [11] is a more complex system for developing malware knowledge based on dynamic analysis in virtualized environments and machine learning techniques to learn models to classify samples. Even though more similar to our work compared to those briefly described so far, it doesn't dealt with automatic sample retrieval, interactive manual analysis and sharing of developed knowledge base.

The works most similar to what we propose are [12], [13], and both regard a framework for malware analysis for Taiwan campuses. The samples they analyze are automatically retrieved using honeynet systems, so they don't generalize the means that can be used to collect samples from distinct sources. They account for information sharing with external entities, but related information are very limited and do not allow a proper comparison with what we propose.

III. Requirements

This section defines the main general requirements for the malware analysis framework, which guide the design of its architecture in next section.

Distinct sources for malware samples [R1]. Being able to feed as many samples as possible is fundamental to get a proper coverage of latest threats. The framework shall allow to input samples coming from a variety of sources, including known malware datasets, honeypots and malware crawlers.

Mostly automatic analysis [R2]. Expecting large amounts of malware to inspect, the analysis of input samples shall be carried out automatically, anyway allowing for intervention by analysts to enhance the quality and the depth of the examination. Samples deserving more thorough investigation shall be pointed out by the framework, with an indication about why such a deepening is advised and a rank on how much worthy this sample is to be further examined.

Intra-CI collaboration [R3]. Manual malware analysis is a required complement to automatic analysis in order to cope with the expected multitude of variants and unknown situation that are likely to be found. The framework shall support the cooperation among malware analysts of a same CI by allowing more authorized users to manually contribute to the analysis of a single specific sample.

Continuous development of the knowledge base [R4]. Since one of the main goals is to generate and maintain an updated knowledge base about cyber threats, a natural requirement regards the capability to continuously increment and refine all the models learned from malware analysis.

Private knowledge base [R5]. Due to high confidentiality needs of CIs in the area of cyber security, the growth of the knowledge base shall go on without relying on any external security provider that would require possibly sensitive information to leave CI's borders.

Inter-CI collaboration [R6]. Even though the previous requirement seems to call for CIs to evolve their knowledge base in total isolation, collaboration among distinct organizations through data sharing is known to be beneficial for improving the cyber security awareness of all participating organizations. The framework shall provide for a highly controlled export of partial views of the knowledge base to external CIs, according to well defined policies, and in turn enable these CIs to deliver their own data share.

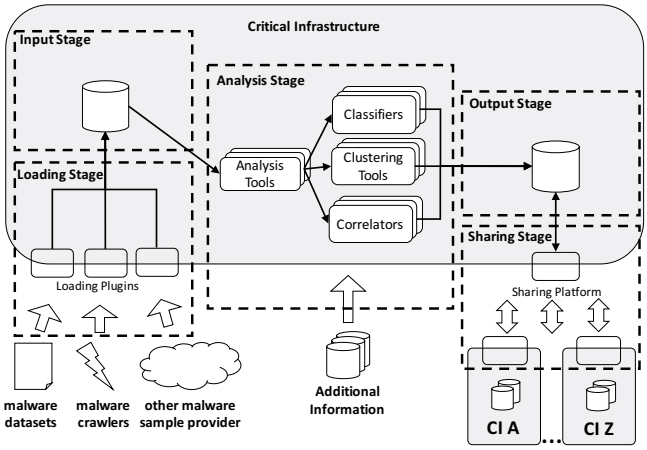

Figure 1. Staged view of the architecture of the malware analysis framework.

IV. Architecture

In this section we describe the architecture for the malware analysis framework. We first show how sample analysis flow is arranged through a staged view of the architecture (§ IV-A). Then a high-level layered view of the architecture is presented (§ IV-B), to highlight main building blocks and interactions of the framework within CI's borders and with external CIs. The details about identified layers are then explained in § IV-C. Finally, a mapping of this architecture to the requirements to meet is presented (§ IV-D).

A. Staged View and Data Flows

We envision the flow of sample analysis as organized in a series of stages, from sample retrieval to information sharing. Figure 1 shows such a staged view of the architecture. The first stage is the *Loading Stage*, where malware samples are gathered from distinct sources such as known datasets, malware crawlers and samples obtained thanks to honeypots. Once loaded, samples are stacked in an *Input Stage* together with a set of metadata related to both the samples themselves and the way they have been retrieved. From the Input Stage, malware can then be injected into the *Analysis Stage*, which takes in charge all the kinds of automatic analyses to be performed. At a very general level, we can structure the Analysis Stage as:

- a set of *Analysis Tools* that examine in details both the content and the sample behavior to produce comprehensive result datasets;
- a set of *Classifiers* in charge of performing advanced elaborations, based on machine learning techniques, on such datasets;
- a set of *Clustering Tools* responsible for grouping samples that share similar characteristics, such as behavior or malware family;

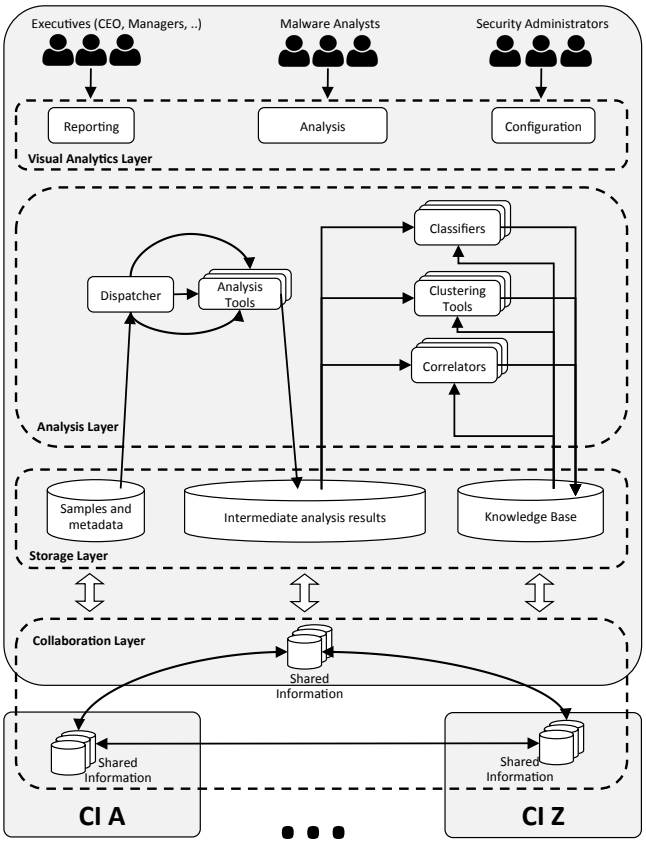

Figure 2. High-level layered view of the architecture of the malware analysis framework, where intra-CI (persons with different roles within the CI) and inter-CI (information sharing among different CIs) interactions are highlighted.

- a set of *Correlators* that retrieve from trusted external sources fresh information about cyber threats, and try to match them with available data resulting from the analyses performed so far.

The results of the Analysis Stage are then pushed to the *Output Stage*, at disposal of authorized analysts for further analysis. More analysts can contribute to the same sample by providing additional information (intra-CI collaboration). Finally, a subset of the data contained in the Output Stage, selected according to properly defined policies, can be moved to the *Sharing Stage*. Data in this stage can be retrieved by external CIs for their own analyses, and the same CIs can also provide their own malware related data. In this way, each participating CI can benefit because the added value resulting from shared information exceeds the value produced by each single CI (inter-CI collaboration).

B. Layered View

Figure 2 presents an high-level layered view of the architecture.

Visual Analytics Layer. Allows authorized users to monitor and control the running tasks within the framework

by means of several graphical user interfaces. Users with different roles can access different information and specific actions. Three roles have been identified: *executives*, *security administrators*, and *malware analysts*. Executives can access *reporting* functionalities to get an overall prospect about framework operational state and knowledge base. Security administrators can supervise and govern every kind of *configuration* related to the framework, including policies that control what information to share. Malware analysts can manually intervene in the *analysis* process and view all the data related to ongoing elaborations and related results. More analysts can contribute additional results for a same malware sample (*intra-CI collaboration*).

Analysis Layer. Contains all the components involved in the automatic elaboration of malware samples performed in the Analysis Stage. It includes an initial *Dispatcher* in charge of sending samples to the right *Tools for Analysis*, depending for example on the file type or on its source. The outputs of these tools are then provided to *Classifiers*, *Clustering Tools*, and *Correlators* by using the Storage Layer. The sequence of elaboration steps that each sample has to pass through is not static, it is instead defined according to the initial dispatching and possibly to some intermediate analysis results. After each elaboration step, the correspondent results are stored in the Storage Layer.

Storage Layer. Includes all the partitions used for storing data: *(i)* one for the Input Stage (*Samples and metadata*), which collects new samples to analyze and related metadata, *(ii)* one for maintaining intermediate data resulting from the Analysis Stage (*Intermediate analysis results*), and *(iii)* one for the Output Stage (*Knowledge Base*).

Collaboration Layer. A view of data contained in the Storage Layer is extracted and used for information sharing with other external CIs (*inter-CI collaboration*). Regardless of the internal architecture used by other CI to produce their own knowledge base, we envision this layer as a cross border tier gluing together different CIs for the sake of controlled sharing of cyber threat knowledge.

C. Layers Details

Each layer is described in more details, according to what is shown in Figure 3.

1) Visual Analytics Layer: the three functionalities of this layer (see § IV-B and Figure 2) are implemented through four *Monitor & Control* (M&C) components.

Input Samples M&C. Allows analysts to load new samples by means of the *Loading Plugins*, including the possibility to manually inject new samples. It also provides access to sample details contained in the Sample and metadata partition. Moreover, such layer implements part of the Analysis functionality of the Visual Analytics Layer.

Analysis M&C. Enables malware analysts to interact with components in the Analysis Layer and, hence, contribute to the analysis process. For example, an analyst could: *(i)* disassemble a sample to bypass anti-disassembling techniques, *(ii)* import external unstructured reports to add external useful information, and *(iii)* enhance sample analysis results by loading into the Storage layer reports and annotations gathered from human analysis.

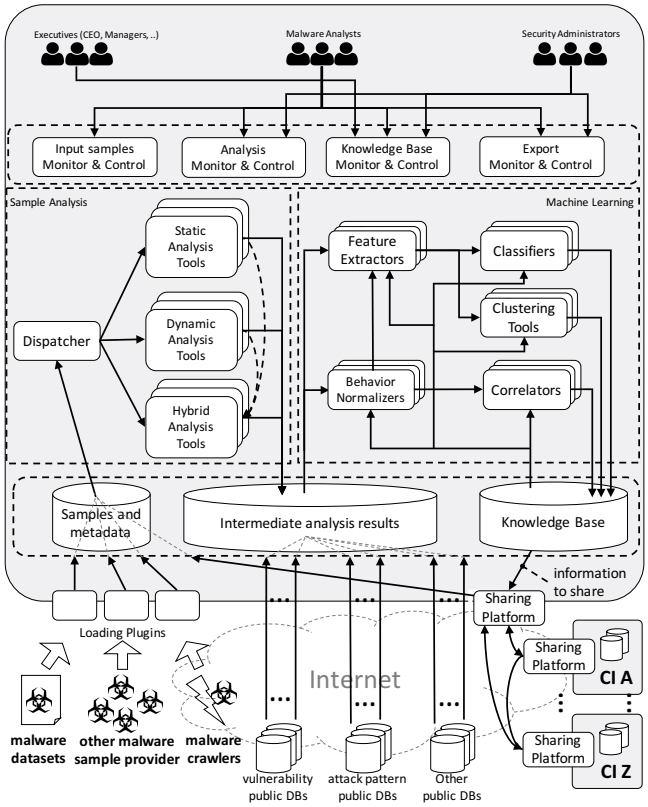

Figure 3. Detailed view of the architecture, where more particulars are presented related to each layer and the way it interacts with other layers, CI's authorized users, and the outside.

Knowledge Base M&C. Provides an interface for accessing results computed by the Analysis stage. This layer is included in the Output stage and supports authorized users by presenting them a *reporting interface*, which contains accurate statistical information.

Export M&C. As part of the Sharing stage, this layer allows security administrators and malware analysts to, respectively, define and implement policies for exporting and sharing partial views of the knowledge base to external CIs. Policy implementations and definitions can be specified through proper *configuration interfaces*.

2) Analysis Layer: consists of all the components involved in sample analyses and comprises the Analysis Stage. This layer is further organized in two sub-layers: the *Sample Analysis*, with tools employing static, dynamic, hybrid analyses, and *Machine Learning*, which contains components able to leverage machine learning techniques to extract new useful sample-related information. Such information are then

collected and stored in the Knowledge Base partition. The components of these two sub-layers are, then, described in the following.

Dispatcher. Automatically retrieves not analyzed samples from the Samples and metadata partition and dispatches them to the proper analysis tool, according to sample characteristics (i.e., a 32-bit portable executable will be forwarded to a dynamic analysis tool able to execute it).

Static, dynamic, and hybrid tools. Receive samples from the dispatcher and perform different types of analyses. While dynamic tools require execution of the input samples, static tools only look at their content. Hybrid tools run analyses based on multiple information, such as the behavior of the samples and their disassembled code and, in addition, can leverage available metadata. These tools can be commercial products, open source software or directly developed by the CI itself. Results are stored into the intermediate results partition or further used as input to hybrid analysis tools.

Feature extractors and behavior normalizers. Features extractors draw values of specific features from the results produced by the analysis tools. The sets of features to extract depend on the downstream classifiers and clustering tools to feed. Behavior normalizers are meant to prepare the input for the correlators by converting to a common format all the data related to sample execution. These data come from both dynamic and static analysis tools, and trusted external sources. In general, extractors and normalizers can be also fed with models and profiles stored in the Knowledge Base.

Classifiers and clustering tools. Classifiers assign samples to predefined learned classes, also defined in the Knowledge Base, on the base of features values. Clustering tools group samples according to their similarity, which enables to isolate specific families of malware and link unknown samples to such families. Each classification and clustering tool leverages a different machine learning technique. Results obtained by these tools are saved into the Knowledge Base and made available for further investigations, including a *rank* regarding whether samples are noteworthy and deserve special attention by analysts. Such a *rank* depends on classification confidence level (the lower the confidence, the higher the attention it requires) and on the extent a sample is marked as an outlier by clustering algorithms. Since the Knowledge Base is continuously updated, classifiers need either to be retrained periodically or to be designed so as to enable online learning.

Correlators. Correlate normalized information related to the behavior of analyzed samples with known malicious behavioral patterns obtained from external sources. The resulting degree of similarities is a valuable result for understanding the nature of a sample. On the other hand, the lack of correlation between information provided by external sources and analyzed samples triggers the need for further analysis by an analyst. Depending on detected correlation degree, a *rank* is associated to each sample to quantify how much worthy it is to be analyzed in more depth.

While there exist several commercial and open source software that can be used for the tools of Sample Analysis sub layer, this is not true for the Machine Learning sub layer, also because of the greater complexity in the structure of inputs to be provided. Thus such software are more likely to be developed ad hoc, possibly by leveraging existing Machine Learning algorithms implementations.

3) Storage Layer: As mentioned before, three distinct storage partitions have been identified.

Samples and metadata partition Stores new samples coming from the Loading plugins. Since samples are collected from distinct sources, they could have different representation formats. Loading plugins handle the format normalization of samples and their related metadata. For example, relevant metadata are the MD5 signature, the list of sources the sample comes from, and its current status in the Analysis stage.

Intermediate analysis results partition Each elaboration step in the Analysis stage produces an intermediate result that is saved into this partition. This latter stores also additional information fetched from the Internet, such as updated publicly known vulnerabilities and attack patterns. In this way, further analyses may leverage intermediate results, obtained in previous elaboration steps, and publicly available security information.

Knowledge Base When the analysis of a sample completes, all the related results and outputs are consolidated in this partition. A partial view of such data is periodically exported to an information sharing subsystem to enable inter-CI collaboration.

4) Collaboration Layer: as discussed in § III, the proposed framework aims to share, under well defined policies, partial views of the Knowledge Base to support inter-CI collaboration. This layer is composed by all the collaborative environment participants' sharing platforms.

Sharing platform Contains a subset of the information that a CI wants to share with other critical infrastructures. The platform should enforce the adoption of policies preventing sensitive information disclosures. Indeed, export data related to tailored malicious samples could represent a concern.

Instead of developing custom applications, Storage and Collaboration Layers can be realized by using either commercial products and open source software. Furthermore, they are more likely to provide interoperability guarantees that would allow to meet inter-CI collaboration requirement more easily.

D. Requirements Satisfaction

Requirement **R1** is satisfied by the plugin-based design of the *Loading Stage*, which allows to include additional sample sources over time by developing the related plugins.

The need for automation reported in requirement **R2** is satisfied by the design of the *Analysis Stage*, where all the

elaboration steps, from sample retrieval to results storage, are executed in pipeline without human intervention. For what concerns the other aspects highlighted by requirement **R2**, *Classifiers*, *Clustering Tools* and *Correlators* provide a rank to signal samples that require further investigation, and the *Analysis M&C* component in the *Visual Analytics Layer* enables malware analysts to manually contribute to the analysis (intra-CI collaboration).

Such component also enables to meet requirement **R3**, since it allows more analysts to collaborate among each other by contributing analysis results for a same malware.

As more samples are analyzed, collected results leads to a continuous enrichment of the knowledge base. Also the models used in the Analysis Layer for classification and clustering are kept updated by employing periodic retraining and online learning techniques. This accounts for requirement **R4**.

The need for keeping the knowledge base as much confidential as possible (**R5**) is satisfied by providing full control on all the interfaces with the outside. Loading plugins can be designed and implemented by embedding anonymization techniques to hide the identity of who is collecting samples. The retrieval of additional public data from the Internet, such as known vulnerabilities, don't pose any further issue. What is shared with other CIs can be highly configured by security administrator using the *Export* component in the *Visual Analytics Layer*.

Finally, the need for inter-CI collaboration (**R6**) is fulfilled by the *Collaboration Layer*.

V. Conclusion

Large organizations like critical infrastructures struggle to find a good balance between full-time in-house expertise on cyber security and outsourced services. Due to the particular nature of Critical Infrastructures that can impact national security, this paper advocates that CIs should be equipped with a domestic knowledge base of cyber threats, supervised by a full-time team of cyber analysts, all this included within well defined internal processes.

The paper presented an architecture for semi-automatic (analysts can intervene when required) collaborative (both intra-CI and inter-CI) malware analysis that CIs can adopt to design and implement their own framework for developing internal cyber threat knowledge base. The contributions of this work are (i) the identification and definition of the main requirements for such a framework, (ii) the explanation of the way malware analysis data flow across the proposed architecture (*staged view*), (iii) the description of how each layer of the architecture works and how they allow to meet defined requirements (*layered view*), and some sketches on the ongoing implementation of a framework prototype.

As a natural future work, we envision to complete prototype implementation and extensively test all its functionalities, and take stock of the experience to review and refine the proposed architecture, deepen implementation and testing issues, and report comprehensive evaluation results related to the actual capabilities of the framework to address the needs of CIs in the field of cyber security.

Acknowledgment

The work is partially supported by a grant of the Italian Presidency of Ministry Council and by the Laboratorio Nazionale of Cyber Security of the CINI (Consorzio Interuniversitario Nazionale Informatica).

References

[1] CNN, "Nearly 1 million new malware threats released every day," 2014. [Online]. Available: http://money.cnn.com/2015/04/14/technology/security/cyber-attack-hacks-security/

[2] E. Chien, "W32.stuxnet dossier," 2010.

[3] R. Lipovsky, "Blackenergy trojan strikes again: Attacks ukrainian electric power industry," January 2016.

[4] G. Lodi, L. Aniello, G. A. Di Luna, and R. Baldoni, "An event-based platform for collaborative threats detection and monitoring," *Information Systems*, vol. 39, 2014.

[5] W. Zhao and G. White, "A collaborative information sharing framework for community cyber security," in *2012 IEEE Conference on Technologies for Homeland Security*, 2012.

[6] S. Alam, R. N. Horspool, and I. Traore, "A framework for metamorphic malware analysis and real-time detection," in *IEEE 28th International Conference on Advanced Information Networking and Applications*, 2014.

[7] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Systems with Applications*, 2016.

[8] S. Alam, R. N. Horspool, and I. Traore, "Mail: Malware analysis intermediate language: a step towards automating and optimizing malware detection," in *Proceedings of the 6th International Conference on Security of Information and Networks*, 2013.

[9] H. Yin and D. Song, *Automatic Malware Analysis: An Emulator Based Approach*. Springer, 2012.

[10] C. A. Borges de Andrade, C. Gomes de Mello, and J. C. Duarte, "Malware automatic analysis," in *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC)*, 2013.

[11] A. Mohaisen, O. Alrawi, and M. Mohaisen, "Amal: High-fidelity, behavior-based automated malware analysis and classification," *Computers & Security*, 2015.

[12] Y.-L. Tsai, L.-Y. Yeh, B.-Y. Lee, and J.-G. Chang, "Automated malware analysis framework with honeynet technology in taiwan campuses," in *Parallel and Distributed Systems*, 2012.

[13] L.-Y. Yeh, Y.-L. Tsai, B.-Y. Lee, and J.-G. Chang, "An automatic botnet detection and notification system in taiwan," in *Proceedings of the 2013 International Conference on Security & Management*, 2013.